University of Groningen

A Mapping Study on Software Artifacts Traceability

Charalampidou, Sofia; Ampatzoglou, Apostolos; Avgeriou, Paris

A Mapping Study on Software Artifacts Traceability: Review Protocol

Sofia Charalampidou
Apostolos Ampatzoglou
Paris Avgeriou

December, 2013

In recent years, mapping studies are increasingly attracting research attention in the field of software engineering. Systematic review and systematic mapping have provided mechanisms to identify and aggregate research evidence. While systematic review has been used to provide a complete and fair evaluation of state of evidence related to a topic of interest (Kitchenham and Charters, 2007), systematic mapping (also known as scoping review) is a more open form of systematic review, providing an overview of a research area to assess the quantity of evidence existing on a topic of interest (Kitchenham and Charters, 2007). The research objectives for mapping studies are, in most cases, high level and include issues, such as identification of research sub-topics, identification of used research methods and possibly, a discussion whether the area under study provides the needed research capacity for a systematic review, which will synthesize data in more detail. Consequently, mapping studies can provide important results and research directions to software engineers by providing an overview of the literature in specific topic areas.

This study aims at summarizing already proposed techniques related to artifact traceability, among and within software development phases. With the term development phases we refer to the parts of a software development lifecycle, as defined in the 12207:2008 ISO/IEC/IEEE standard, i.e. Requirements Analysis, Architecture Design, Detailed Design, Construction, Integration, and Quality Testing (12207 IEEE standard). This standard introduces a process framework for describing the development lifecycle of the software part of a system and is composed by the six abovementioned development phases. Additionally, according to the same standard a development phase consists of several activities (e.g. the Requirements Analysis phase may consist of representing system requirements using natural language and writing use cases). In this document we will refer to such activities as "development activities".

In Section 1, we will provide background information on artifact traceability. In addition to that, we will describe the motivation for conducting this mapping study and a discussion on the reasons for selecting to conduct a mapping study, rather than a systematic literature review. Next, in Section 2, we will present the systematic mapping protocol, whereas in Sections 3 and 4 we will present the protocol validation process and the plan for presenting the results of our study, respectively.

1. Introduction

Traceability is a term used in the software engineering domain for referring to the potential of creating links between software artifacts. The process of achieving after-the-fact traceability is found in literature with the term **trace recovery** and it is defined as the "approach to create trace links after the artifacts that they associate have been generated or manipulated" (Cleland-Huang et al, 2012). In addition, there is also the so-called **trace capture** or real-time traceability, i.e. linking artifacts while specifying and developing the system in a forward engineering manner. Trace capture is defined as a "particular

approach to trace creation that implies the creation of trace links concurrently with the creation of the artifacts that they associate" (Cleland-Huang et al, 2012).

In addition to the abovementioned definitions, Sommerville and Kotonya (Sommerville and Kotonya, 1998) suggest that trace recovery, can be further divided into four different types, with requirements as starting or ending point of traceability. These are:

a) Backward-from traceability: Links requirements to their sources which could be other documents.
b) Forward-from traceability: Links requirements to the design and implementation components.
c) Backward-to traceability: Links design and implementation components back to the requirements.
d) Forward-to traceability: Links other documents (e.g. operation manuals describing the system functionality) to relevant requirements.

1.1 Artifact Traceability Classification Schema

In this study we aim at collecting trace recovery (after-the-fact) approaches, regardless of their direction (forward or backward), through a systematic mapping study. This means that we will focus on approaches that create traces after the software artifacts of interest have been created, and that the traces will be possible to be created either forwardly or backwardly. In contrast to (Sommerville and Kotonya, 1998) this study will not be restricted only to tracing approaches connecting requirements with other software artifacts but will concern all software artifacts. However, we note that the scope of this study is only on trace recovery, as defined in (Cleland-Huang et al, 2012), and will not investigate the research state of the art on trace capture.

One of the main objectives of a mapping study is to propose a classification schema regarding research efforts on the investigated topic. The expected classification schema for traceability approaches will be based on (Sommerville and Kotonya, 1998), with two potential main points of deviation:

- In this study we aim at including the possibility of creating traces between artifacts produced in the same development phase, but within different development activities. This class of traceability approaches, is not covered by (Sommerville and Kotonya, 1998), in the sense that it only divides tracing approaches, to forward and backward approaches. The existence of such approaches in literature has been reported in (Borg et al. 2013), where the authors conducted a systematic mapping study on Information-Retrieval-based trace recovery. Thus, neglecting them would limit down the scope of this study and the expected classification schema.
- Furthermore, we propose an additional level in the classification schema, for further differentiation among approaches that trace artifacts of different development phases. Intuitively,

it is expected that most approaches will provide traces between adjacent development phases, i.e. development steps which follow one another without intermediate steps in between (e.g. architecture design and detailed design). However, pilot searches indicate the existence of traceability approaches between non adjacent development phases (e.g. requirements and code). Such approaches might be useful in project specific cases when some development phases are omitted.

Thus, the first level of categorizing the tracing approaches will be done with respect to the development phases that the linked artifacts belong to. Whereas, traceability approaches linking software artifacts of different development phases are classified based on whether they are forward approaches (e.g. from requirements to architecture) or backward approaches (e.g. from architecture to requirements) (Sommerville and Kotonya, 1998), and further categorized based on whether or not they create traces between software artifacts of adjacent development phases. The classification of the potential categories of existing traceability approaches as described above is also presented in Figure 1.

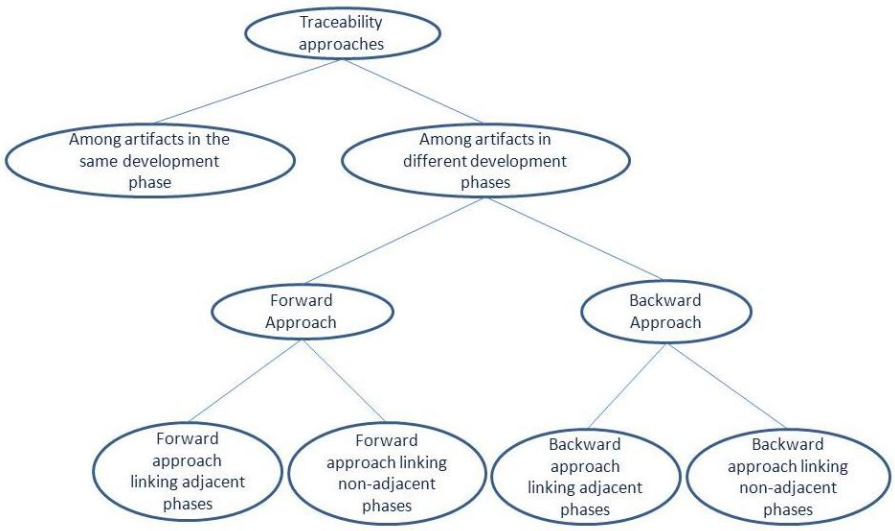

Figure 1. Classification of potential categories of artifacts tracability approaches.

Next, we present some examples of the classes of the classification schema presented above:

- **Traceability approaches among artifacts of the same development phase within different development activities**: *An approach that creates traces between natural language requirements and use case scenarios. Both artifacts belong to the Requirements Analysis development phase,*

but the first one would belong to the activity "Elicit stakeholders' needs" while the second one to the activity "Specify use-cases".

- **Forward traceability approaches among artifacts of different adjacent development phases**: *An approach creating links between the requirements and the architecture design, or the architecture design and the detailed design, or the detailed design and the implementation.*
- **Forward traceability approaches among artifacts of different non adjacent development phases**: *An approach creating links between the requirements and the detailed design (requirements and detailed design have as intermediate step architecture design).*
- **Backward traceability approaches among artifacts of different adjacent development phases**: *An approach creating links between the implementation and the detailed design (starting from the implementation).*
- **Backward traceability approaches among artifacts of different non adjacent development phases**: *An approach creating links between the detailed design and the requirements, starting from the detailed design, (requirements and detailed design have as intermediate step the architecture design).*

1.2 Benefits from Creating Artifact Traces

According to the literature, the creation of traces among software artifacts can be proven to be beneficial in several ways. In (Antoniol et al. 2002) the authors investigate the use of traces between free text documentation and code either during the development or maintenance cycle. The potential benefits of such an approach include better program comprehension, easier maintenance activities, capability to assess the completeness of the product based on the traced requirements, impact analysis and a good foundation for reusing existing software. Additionally in (De Leon and Avlves-Foss, 2013) it is stated that traceability relationships among artifacts enable incremental verification and validation of correctness and dependability properties, as well as analysis of compliance with existing standards and certifications during the lifespan of the project.

In addition to the aforementioned benefits, we present a number of use cases where the creation of traces among software artifacts could resolve commonly occurring problems.

- **Understanding System Decomposition**: The system decomposition is usually more clear in the architectural level rather than in the design or code levels. The creation of traces between the architectural level and the code and the detailed design levels helps to transfer the decomposition knowledge to all levels.
- **Understanding the Functionality of System Modules**: As mentioned above the system decomposition is usually easier to understand in the architectural level, while the quality

characteristics and the functionality of the system are captured through functional and non-functional requirements in the requirements level. Thus, linking the requirements and the architectural level provides the possibility to map the decomposed system modules to requirements.

- **Performing Corrective Maintenance Requests** (ISO/IEC 14764:2006): In the example of a potential logical defect, which can easily be connected with a specific requirement, the existence of end to end traces (from requirements to code) can indicate the classes implementing this particular requirement to which debugging should focus.
- **Performing Adaptive Maintenance Requests** (ISO/IEC 14764:2006):In the case of a potential need for extending the system functionality (by adding a new requirement or changing an existing one), an indication on which components and classes are implementing this functionality can be considered as useful input regarding which parts of the system should most probably get modified. Also links among requirements, between requirements and architectural components, and among architectural components could be used for performing change impact analysis on the indirect changes that might be caused by cycles in the dependency tree of the system.
- **Performing Perfective Maintenance Requests** (ISO/IEC 14764:2006): In the case that a quality improvement of our system is required, it can be proven beneficial to identify the parts of the code that should have better quality, i.e. parts of the system that change more frequently. The requirements are artifacts that can be characterized as change prone or stable and thus the links can provide guidance on which design and code parts where perfective maintenance should focus.
- **Performing Preventive Maintenance Requests** (ISO/IEC 14764:2006): In the case of searching for existing defects that have not been identified yet, it could be useful to investigate parts of the code that show increased complexity. At the requirements level the complexity of a requirement is usually assigned with the form of a risk level. A risk prone requirement is more probable to create defects in the future. Thus the traces linking the requirements to the code can provide guidance on which design and code parts the preventive maintenance should focus on.

1.3 Need for the Mapping Study

A common challenge in industry is that software artifacts produced during a project's lifecycle are not linked with each other or the traces between them are not sufficient for the existing needs. In such cases, as described above, a trace recovery process could be very beneficial in industrial practice. This study will investigate the state of the art with respect to the existing trace recovery approaches, aiming to explore what kind of approaches exist and how they are currently used. More specifically, we expect the following contributions:

c1: **Create a classification schema for adequately classifying artifact traceability approaches**. This specific mapping study outcome is expected to validate and/or refine the classification schema of traceability approaches, presented above. The existence of such a classification would help in identifying gaps in the literature, research trends, while it could also be beneficial in an industrial context, since it could be used as a catalog that practitioners could use for choosing existing solutions. In addition to that, the classification schema will be expanded with an additional level describing the most commonly linked software artifacts and consequently the linked development phases. Furthermore, potential differentiations in the definition of the generic term *traceability* will be assessed, among different artifact traceability approaches.

c2: **Characterize each class of proposed classification schema in terms of research intensity and level of evidence**. The outcome of this process is expected to map each class with two attributes: (a) the number of studies that have already used/investigated/evaluated or proposed traceability approaches of the class, and (b) an assessment of the level of empirical evidence that the studies, dealing with the corresponding class, provide. This characterization is also expected to further investigate possible gaps and trends in the research state of the art, and guide practitioners to the use of the most established approaches.

c3: **Create a classification schema for adequately classifying possible constraints of the artifact traceability approaches**. This additional classification schema aims at investigating the potential need for applying certain transformations to the software artifacts in order to use an identified tracing approach. An example of a constraint could be the use of a specific ADL for enabling the tracing of an architectural artifact. Thus, using this traceability approach in a system not designed with a specific ADL, should undergo some transformations before the application of the traceability approach. Mapping traceability approaches to certain classes of this classification schema, will provide both researchers and practitioners guidelines for more accurately identifying the traceability approach that they could use, not only depending on the classification proposed in c1, but also in terms of limitations of the approach itself.

1.4 Related Work

In this section we present related work and therefore summarize and compare the goal of this study to other systematic literature reviews or a mapping studies on the domain of software artifact traceability.

Torkar et al. have conducted a systematic literature review on requirements traceability. The study considers primary studies during the period 1997 to 2007 and aims at answering two main research questions: (a) regarding the existing definitions of the requirements' traceability, and (b) regarding the

existing requirements' traceability techniques, their challenges and the related tools found in literature (Torkar et al., 2012).

Later, Borg et al. conducted a systematic mapping study on information retrieval (IR)-based trace recovery approaches. The scope of the study is limited down focusing only on traceability approaches of natural language (NL) software artifacts, during the years 1999 to 2011. The research questions that are investigated during the study consider (a) the identification of the most frequent IR approaches for tracing NL software artifacts, (b) the types of the artifacts that are most commonly linked, and (c) the level of evidence during the evaluation of these approaches (Borg et al. 2013).

Although both the aforementioned studies investigate trace recovery approaches, similar to this mapping study, they have several significant differentiation points compared to our mapping study:

- our approach is broader in scope, in the sense that we are interested in classifying traceability approaches linking any types of software artifacts, without limiting the scope only to NL (Borg et al. 2013) or requirements trace recovery (Torkar et al., 2012),
- our approach is differentiating between the development phases of architectural and detail design, which were merged by Borg et al. (Borg et al. 2013),
- our study will contain studies until 2013, i.e. two additional years than the most recent studies (Borg et al. 2013).

1.5 Why a Mapping Study and not a Systematic Literature Review?

According to Kitchenham et al. there are several differences between a mapping study and a systematic literature review (SLR). Whereas the most important criterion for selecting the most appropriate form of a secondary study is its goal and scope (Kitchenham et.al. 2011). The goal of a mapping study is the classification and thematic analysis of the literature on a specific topic. The scope of the study is usually broad and although all papers related to a topic are considered as primary studies, only information relevant to the classification is collected. On the other hand an SLR aims at "*identifying best practice with respect to specific procedures, technologies, methods or tools by aggregating information from comparative studies*" (Kitchenham et.al. 2011). The scope of an SLR is more focused and the papers included are usually empirical papers related to a specific research question. Additionally, the information extracted from each paper is usually about individual research outcomes.

As already presented above, this study aims at proposing and evaluating a classification schema, based on existing literature on the topic of software artifacts traceability, which is substantially broad. Thus, based on the definition of the two methods, and our goal and scope, it is more appropriate to conduct a mapping study instead of an SLR .

2. Review Protocol

This section describes the protocol of the proposed systematic mapping study. A protocol constitutes a pre-determined plan that describes research questions and how the mapping study will be conducted. To conduct this mapping study we will follow the process proposed by Petersen (Petersen et al., 2008). The essential process steps of a systematic mapping study are: (a) definition of research questions, (b) conducting the search for relevant papers, (c) screening of papers, (d) keywording of abstracts, and (e) data extraction and mapping.

2.1 Research Questions

In this study, we plan to summarize existing techniques that have been used for artifacts traceability. The research questions have been identified using a Goal-Question-Metrics (GQM) approach (Basili et al., 1994). GQM defines a top down approach that aims at developing meaningful metrics. The approach introduces three levels:

(a) Conceptual level (goal): "A goal is defined for an object for a variety of reasons, with respect to various models of quality, from various points of view and relative to a particular environment" (Basili et al., 1994).

(b) Operational level (question): "A set of questions is used to define models of the object of study and then focuses on that object to characterize the assessment or achievement of a specific goal" (Basili et al., 1994).

(c) Quantitative level (metric): "A set of metrics, based on the models, is associated with every question in order to answer it in a measurable way" (Basili et al., 1994).

The goal of this mapping study in terms of GQM is as follows:

Object: Software artifacts traceability

Purpose: Analyze and characterize

Issues:

i. Create a classification schema for adequately classifying artifact traceability approaches (see c1 in Section 1.3).

ii. Characterize each classification class in terms of research intensity and level of evidence (see c2 in Section 1.3).

iii. Create a classification schema for adequately classifying possible constraints of the artifact traceability approaches (see c3 in Section 1.3).

Viewpoint: Software engineers and researchers

According to the abovementioned goals, and the expected contributions (see section 1.3) we extracted six Research Questions (RQ), as follows:

RQ_1: *What is an adequate classification schema for classifying the existing artifact traceability approaches?*

RQ_2: *How to expand the proposed classification schema so as to further classify artifact traceability approaches, with respect to the linked development phases and software artifacts?*

RQ_3: *What are the different definitions of traceability among studies that propose different artifact traceability approaches?*

RQ_1, RQ_2 and RQ_3 deal with c1, i.e. "*Create a classification schema for adequately classifying artifact traceability approaches"*.

RQ_4: *Which of the identified classes of artifact traceability approaches have been more frequently studied in the literature?*

RQ_5: *What is the level of empirical evidence on these approaches in practice?*

RQ_4 and RQ_5 deal with c2, i.e. "*Characterize each classification class in terms of research intensity and level of evidence"*.

RQ_6: *What is an adequate classification schema for classifying artifact traceability approaches, with respect to the possible kinds of constraints that they may have?*

RQ_6 deals with c3, i.e. "*Create a classification schema for adequately classifying possible constraints of the artifact traceability approaches"*.

Generally, mapping studies aim to provide researchers with information about the type and amount of research available and do not assess the outcome of primary studies. However, according to (Kitchenham et al., 2011) although mapping studies and systematic reviews have rather different goals, there is often an overlap, see for example (Kitchenham et al., 2010) which although the paper is mainly a mapping study also includes an assessment of the outcomes of some papers in one of the categories. The metrics used for answering the abovementioned research questions are referenced in section 2.5.

2.2 Search Procedure

The search procedure of a systematic mapping study aims at identifying as many primary studies related to the research questions as possible, using an unbiased search strategy. In order to ensure high relevance of the extracted primary studies with the subject of this research, before starting the automated search of primary studies, we will apply a manual search process aiming at (a) ensuring the accuracy of the automated search process, and (b) validating the accuracy of the search string used for the automatic search process. This manual search will be based on some sample primary studies that we will collect from literature sources, where relevant studies on the specific topic of interest are usually published.

These sources shall be relevant to the domain of interest, so that the relevance of the returned studies to be as high as possible. Then, a comparison among the results of the manual and automatic search process will indicate whether the search string is capable to return relevant studies based on the research questions.

This study focuses on trace recovery. In that sense we believe that relevant studies can be identified in venues related to the maintenance phase, when after-the-fact traceability is more probable to apply, and reverse engineering. To the best of our knowledge, the most important venues in these domain are: Journal of Software Maintenance and Evolution (JSME), International Conference on Software Maintenance (ICSM), European Conference on Maintenance and Reengineering (CSMR), and Working Conference on Reverse Engineering (WCRE). In addition to the relevant venues with our topic, we have decided to perform a manual search in the top two journals and conferences, in the domain of software engineering (TSE and TOSEM, and ISCE and FSE accordingly). However, since the amount of papers in these venues is considerably high, we have selected to limit the manual search starting from January 2011.

Based on the manual search, we will refine the automated search process using generic digital libraries, such as:

- IEEE Digital Library
- ACM Digital Library
- Springer Link
- Science Direct
- Willey

The purpose of a systematic mapping is to conduct a review of relevant studies to assess the quantity of evidence existing to address the research questions. The process needs to be rigorous and unbiased and it often involves a wide coverage of sources, such as online databases, journals and conferences. In order to minimize bias and to maximize the number of sources examined, a pre-defined strategy to identify potential primary studies is required. In the venues where we will conduct an automatic search, the search string will consist of four main parts: *tracing* **AND** *artifact* **AND** *software* **AND** *retrieval*. The string that will be created will be searched in the papers' title, abstract and keywords. For each query part, a list of alternate keywords has been used and connected through logical **OR** to form a more expressive query, as follows:

(trace OR tracing OR traceability OR link OR links)
AND
(requirement OR specification OR document OR design OR code OR test OR defect OR artefact OR artifact OR feature OR concept OR concern)
AND
(software OR program)
AND
(recovery OR retrieval)

The outcomes of the automated search are highly dependent on the quality of the search string used, and thus several refinements are needed before the search string can be considered as well defined. For this reason the search string will be validated while being used in a subset of the venues defined in the protocol. The studies fetched from this search will be analyzed aiming at verifying if they are in accordance to the objective of the SLR and the main research questions. In case we discover that the search string can be improved, changes must be applied and the process much be repeated.
All the results of this research will be stored using the JabRef software, an open source bibliography reference manager. The details of the papers (i.e., title, author(s), abstract, keywords, year of publication and the name of the data source) will be directly exported from the digital libraries to JabRef, using a second reference management tool, i.e. Zotero.

2.3 Study Selection (Screening Papers)

The papers that are selected as primary studies in the review must be relevant to trace recovery approaches . In line with (Dyba and Dingsoyr, 2008), there are three stages of filtering the article set to produce the primary study data set. The search process, described in Section 2.2, will return a set of candidate primary studies. On the completion of that phase, the article set will go through two phases of manual inspection. In the first one the criteria will be applied on each article's title, abstract/conclusions, while on the second phase they will be applied on each article's full text. The inclusion/exclusion criteria that will be used in every stage are listed below:

- Inclusion criteria:
  - The study introduces a trace recovery approach
  - The study uses a trace recovery approach
  - The full text of the study is available in English
- Exclusion criteria:
  - The study does not comply to the inclusion criteria(or)
  - The study introduces an approach for tracing the same artifacts across versions (or)

- The study introduces a traceability approach without stating explicitly the software artifacts or development phases that are linked (or)
- The study is an editorial, position paper, keynote, opinion, tutorial, poster or panel(or)
- The study is a previous version of a more complete paper about the same research (or)

Every article selection phase will be handled by the first author and possible doubts will be resolved by the second and third authors. For each data source mentioned in Section 2.2, we will document the number of papers that will be returned. Also, we will record the number of papers left for each venue after primary study selection on the basis of title and abstract. Moreover, the number of papers finally selected from each source will be recorded. The venues to be searched are shown in the table below.

Name of Digital Library	Papers returned	Papers filtered
IEEE Digital Library		
ACM Digital Library		
Springer Link		
Science Direct		
Wiley		

2.4 Keywording of Abstracts (Classification Scheme)

In (Petersen et al., 2008) the authors propose keywording of abstracts as a way to develop a classification scheme, if existing schemes do not fit, and ensure that the scheme takes into account the identified primary studies. In this study, we plan to classify artifact tracing approaches with respect to:

- Whether they link artifacts belonging in the same development phase or not.
- Their direction (backward or forward).
- Weather they link artifacts belonging in adjacent or non-adjacent phases.
- The level of empirical evidence that is provided regarding the effectiveness of the proposed traceability approach.
- The existing constraints that apply in a traceability approach.

In order to ensure the accuracy of the process, we decided to apply the keywording technique to the articles' full texts. So, the researcher will read the complete manuscripts and identify the development phases that the method deals with.

2.5 Data Extraction and Mapping

During the data collection phase, we will collect a set of variables that describe each primary study. Data collection is going to be handled the PhD Student and possible conflicts will be resolved by her supervisor. For every study, we extracted assigned values to the following attributes:

[A1] Author
[A2] Year
[A3] Title
[A4] Source
[A5] Venue
[A6] Type of Paper (conference / journal)
[A7] Keywords
[A8] Name of the proposed tracing approach
[A9] Starting Development Phase (based on 12207:2008 IEEE standard)
[A10] Ending Development Phase (based on 12207:2008 IEEE standard)
[A11] Trace Direction (forward / backward)
[A12] Linking Adjacent development phases (y/n)
[A13] Starting Trace Artifact
[A14] Ending Trace Artifact
[A15] Level of Evidence (Alves et al 2010)
[A16] Approach Constraints
[A17] Traceability Definition

Attributes [A1] – [A7] are going to be used for Documentation reasons. All other variables are going to be used for answering a corresponding research question. Concerning [A13], Kitchenham and Charters (Kitchenham and Charters, 2007) proposed five levels of study design in software engineering. In this study we will use a revised version of the above-mentioned levels of evidence as described by Alves et al. (Alves et al 2010):

1: No evidence.
2: Evidence obtained from demonstration or working out toy examples.
3: Evidence obtained from expert opinions or observations.
4: Evidence obtained from academic studies (e.g., controlled lab experiments).
5: Evidence obtained from industrial studies (e.g., causal case studies).
6: Evidence obtained from industrial evidence.

The mapping between attributes and research questions is provided in the following table, accompanied by the synthesis or analysis methods used on the data.

Research Question	Variables Used	Presentation Method
RQ_1	[A8] – [A12]	Classification Schema
RQ_2	[A9] – [A10] [A13] – [A14]	Updated Classification Schema, by two additional levels
RQ_3	[A17]	Count of different definitions Similarities/Differences among traceability definitions
RQ_4	[A8] – [A14]	Frequency Tables Bubble Charts ([A9], [A10], Count) Bubble Charts ([A13], [A14], Count) Bubble Charts ([A11], [A12], Count) Bubble Charts (all possible combinations of [A9]-[A10], [A11], Count) Bubble Charts (all possible combinations of [A9]-[A10], [A12], Count)
RQ_5	[A9] – [A10] [A13] – [A14] [A15]	Descriptive Statistics Bubble Charts ([A9], [A10], AVG[A15]) Bubble Charts ([A13], [A14], AVG[A15]) Bubble Charts ([A11], [A12], AVG[A15]) Bubble Charts (all possible combinations of [A9]-[A10], [A11], AVG[A15]) Bubble Charts (all possible combinations of [A9]-[A10], [A12], AVG[A15])
RQ_6	[A16]	Classification Schema Frequency Tables Crosstabs ([A16], [A9], A[10]) Crosstabs ([A16], [A13], A[14])

3. Protocol Validation

The systematic review is validated in three parts:

- The pilot – testing the process:

- Researchers use a subset of resources to test the process. Problems in replicating the process are identified, the process is refined accordingly.
- Gaps in searches are identified and search terms and resources are changed to include missing papers.
- Data extraction: reliability of how to extract details from papers is tested. An independent researcher (not involved in the pilot) is given a set of accepted papers and asked to fill in the data extraction form.

❖ External reviewers will review the protocol.

4. Report the Review

When reporting our study, we will a) discuss findings (description of primary studies and results of any quantitative summaries), b) present a discussion (principal findings, strengths and weaknesses of the review, and the meaning of findings, such as applicability of findings, benefits, adverse effects and risks), c) discuss conclusions, including recommendations, such as practical implications for software development, implications for practitioners, and unanswered questions and implications for future research, and d) elaborate on limitations of the review.

References

N. Aizenbud-Reshef, B. T. Nolan, J. Rubin, Y. Shaham-Gafni, "Model traceability", *IBM Systems Journal*, IBM, 45 (3), pp. 515-526, July 2006.

V. Alves, N. Niu, C. Alves, G. Valenca, "Requirements engineering for software product lines: a systematic literature review", *Information and Software Technology*, Elsevier, 52 (8), pp. 806-820, August 2010.

G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, e. Merlo, "Recovering traceability links between code and documentation", *IEEE Transactions on Software Engineering*, IEEE Computer Society, 28 (10), pp. 970-983, October 2002.

V. Basili, G. Caldiera, D. Rombach, "The Goal Question Metric Approach", *Encyclopedia of Software Engineering*, John Wiley & Sons, pp. 528-532, 1994

M. Borg, P. Runeson, A. Ardö, "Recovering from a decade: a systematic mapping of information, retrieval approaches to software traceability", *Empirical Software Engineering*, Springer, accepted for publication, doi={10.1007/s10664-013-9255-y}.

J . Cleland-Huang, O. Gotel, A. Zisman, "Software and systems traceability", Springer, London, 2012.

D.C. de Leon, J. Alves-Foss, “Hidden Implementation Dependencies in High Assurance and Critical Computing Systems”, *IEEE Transactions on Software Engineering*, IEEE Computer Society, 32(10), pp.790-811, October 2006

T. Dyba and T. Dingsoyr, “Empirical studies of agile software development: A systematic review”, *Information and Software Technology*, Elsevier, 50 (9-10), pp. 833-859, August 2008.

IEEE™ 12207-2008 “Standard for Systems and Software Engineering - Software Life Cycle Processes”, IEEE Computer Society, January 2008.

IEEE™ 14764-2006 “Software Engineering - Software Life Cycle Processes - Maintenance”, IEEE Computer Society, September 2006.

B. A. Kitchenham, “What's up with metrics? A preliminary mapping study”, *Journal of Systems and Software*, Elsevier, 83 (1), pp. 37-51, January 2010.

B. A. Kitchenham, D. Budgen, O. P. Brereton, “Using mapping studies as the basis for further research: A participant-observer case study”, *Information and Software Technolology*, Elsevier, 53(6), pp. 638-651, June 2011.

B. Kitchenham and S. Charters. “Guidelines for performing systematic literature reviews in software engineering”, *Technical Report EBSE 2007-001*, Keele University and Durham University, 2007.

K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, “Systematic mapping studies in software engineering”, *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, British Computer Society, pp. 68-77, June 2008.

I. Sommerville, G. Kotonya, “Requirements Engineering: Processes and Techniques”,
John Wiley & Sons, New York, 1998.

R.Torkar, T. Gorschek, R. Feldt, M. Svahnberg, U. Akbarraja, K. Kamran, “Requirements Traceability: A Systematic review and Industry Case Study”, *International Journal of Software Engineering and Knowledge Engineering*, World Scientific Publishing, 22(03), pp. 385-433, May-2012.